Programación Funcional Avanzada

Prolegómenos

Ernesto Hernández-Novich
<emhn@usb.ve>

Universidad "Simón Bolívar"

UNIVERSIDAD SIMÓN BOLÍVAR

¿Por qué Haskell?

Porque Scheme y OCaML lo fueron en su momento

- La expresividad aumenta la productividad.
  - Núcleo del lenguaje pequeño, pero flexible.
  - Código conciso acelera el desarrollo.
  - Compilado e interpretado – lo mejor de ambos mundos.
- Más fácil comprender y mantener el código.
  - Incluso las librerías complejas son comprensibles.
  - Lo conciso deja espacio para documentar mejor.
- Puede aumentar la robustez de un sistema.
  - Sistema de tipos detecta defectos a tiempo de compilación.
  - Estilo funcional permite mejores técnicas de prueba.
  - Paralelismo sin alterar la semántica.
  - Concurrencia resistente a condiciones de carrera.

Estable y sólido para producción

UNIVERSIDAD SIMÓN BOLÍVAR

¿Y para qué esta materia?

- Para aprender a aplicar Haskell de manera práctica
  - Haskell es un vehículo para investigación – reflejo en como se enseña.
  - Vamos a estudiarlo con perspectiva práctica.
- Para aprender técnicas de programación.
  - Nuevas, sorprendentes y efectivas.
  - . . . y algunas se quedarán por fuera.
- Para dejar de lado la frustración de otros lenguajes.
  - No hace falta un nuevo lenguaje – extender Haskell.
  - No hace falta cambiar lenguajes – aprovechar librerías.
  - Las especificaciones "corren" si son consistentes.

Para **disfrutar** la programación

UNIVERSIDAD SIMÓN BOLÍVAR

¿Qué se espera del estudiante?

- Supongo que han usado Haskell en CI-3641/CI-3661.
- Hoy voy a repasar algunos conceptos básicos –
  si no los estudiaron, es conveniente que los repasen.
- Más o menos un tema por clase – transcriban el código.
- La evaluación es completamente práctica
  - Cinco o seis tareas (80 %) individuales –
    en forma *Literate Haskell* (LaTeX con Haskell)
  - **Su** proyecto (20 %) en pareja –
    con presentación y demostración.

Aproveche el proyecto para
explorar sus áreas de interés.

UNIVERSIDAD SIMÓN BOLÍVAR

¿Qué se espera del estudiante?

- Con el desarrollo de los temas habrá un componente transversal de exploración de las librerías.
  - "Hoy estudiaremos foo – voy a aprovechar la librería bar"
  - Supongo que Uds. irán a leer la documentación de las librerías.
- Si instalan una librería, se incluye documentación en HTML.
- Hoogle – Buscador especializado
- Haskell Cafe – es una lista de correos – yo también leo, just sayin'
- Haskell WiKi

UNIVERSIDAD SIMÓN BOLÍVAR

El ambiente de trabajo

- Editor de texto para programación.
  - Yo uso VIM con Haskell Mode.
  - Algunas almas en pena usan EMACS.
  - The Walking Dead eat brains looking for an editor.
  - LEKSAH es un IDE para Haskell escrito en Haskell – **no lo uso, no sé cómo funciona** YMMV.
- Algún controlador de versiones
  - cp es a un VCS lo que la pizza a una dieta – Dropbox es extra tocineta.
  - Al VCS no le importa – usen Git, SVN, ...
  - Pero DARCS está escrito en Haskell.
  - DARCS es equivalente a Git – así no se ofenden los fanboys.
- Yo uso GHC 7.6.3 por razones prácticas – eviten 7.8 por el momento

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo encontrar las librerías?

Con mucho cuidado

- Debian Jessie (¡nueva!) y derivados Debian "decentes"

```
apt-cache search libghc-
```

  - Sufijo `-dev` es la librería.
  - Sufijo `-doc` es la documentación HTML.
  - Sufijo `-prof` sólo se instala para *profiling*.

- Cualquier otra plataforma o para librerías que no están en Debian

```
cabal update
cabal install foo
```

  - Instalará una copia *privada* de la librería y todas sus dependencias.
  - Si la dependencia está en Debian, pero Uds. no la instalaron previamente, **no** lo notará.

- Necesitaremos `cabal` para tres temas – el resto listo en Debian.

Compresión

No tiene pérdida...

- *Run Length Encoding* – secuencia de repetidos sustituido por un testigo y su cantidad

```
> encode "quuuuuuuuxxxx"
[(1,'q'),(8,'u'),(4,'x')]
> decode [(1,'q'),(8,'u'),(4,'x')]
"quuuuuuuuxxxx"
```

- Implantación genérica para trabajar sobre listas.
- `encode` – comprime la lista.
- `decode` – expande lo comprimido.

UNIVERSIDAD SIMÓN BOLÍVAR

Criptopoesía

USB Internal Programming Contest 2010

- Problem D – Decode that poem

Entrada	Salida
Hey g**o**od la**w**yer **a**s I p**r**eviously p**r**e**v**iewed **y**am d**o**es a so**u**p	How are you

  - Cada párrafo de entrada, una línea de salida.
  - Cada palabra por línea de entrada, una letra en la salida.

UNIVERSIDAD SIMÓN BOLÍVAR

Criptopoesía

USB Internal Programming Contest 2010

- Problem D – Decode that poem

Entrada	Salida
Hey g**o**od la**w**yer **a**s I p**r**eviously p**r**e**v**iewed **y**am d**o**es a so**u**p	How are you

  - Cada párrafo de entrada, una línea de salida.
  - Cada palabra por línea de entrada, una letra en la salida.
- Mente Funcional
  - ¿Puedo hacer una transformación global?
  - ¿Puedo dividir esa transformación en partes?
  - Iteraciones implícitas – `map`, `fold`, ...

UNIVERSIDAD SIMÓN BOLÍVAR

Criptopoesía

USB Internal Programming Contest 2010

- Problem D – Decode that poem

Entrada	Salida
Hey g**o**od la**w**yer **a**s I p**r**eviously p**r**e**v**iewed **y**am d**o**es a so**u**p	How are you

  - Cada párrafo de entrada, una línea de salida.
  - Cada palabra por línea de entrada, una letra en la salida.
- Mente Funcional
  - ¿Puedo hacer una transformación global?
  - ¿Puedo dividir esa transformación en partes?
  - Iteraciones implícitas – `map`, `fold`, ...
- Transformaciones puras primero – interacción después.

UNIVERSIDAD SIMÓN BOLÍVAR

Jingle Composing

ACM ICPC2009 Latin American Regionals

- Problem J – Jingle Composing
  - `W` $= 1$, `H` $= 1/2$, `Q` $= 1/4$, `E` $= 1/8$, `S` $= 1/16$, `T` $= 1/32$ y `X` $= 1/64$.
  - Sea una línea de la forma

    `/HH/QQQQ/XXXTXTEQH/W/HW/`
  - ¿Todos los compases tienen la misma duración?

Jingle Composing

ACM ICPC2009 Latin American Regionals

- Problem J – Jingle Composing
  - `W` $= 1$, `H` $= 1/2$, `Q` $= 1/4$, `E` $= 1/8$, `S` $= 1/16$, `T` $= 1/32$ y `X` $= 1/64$.
  - Sea una línea de la forma

    `/HH/QQQQ/XXXTXTEQH/W/HW/`
  - ¿Todos los compases tienen la misma duración?
- Mente Funcional
  - ¿Puedo hacer una transformación global?
  - ¿Puedo dividir esa transformación en partes?
  - Iteraciones implícitas – `map`, `fold`, ...

UNIVERSIDAD SIMÓN BOLÍVAR

Jingle Composing

ACM ICPC2009 Latin American Regionals

- Problem J – Jingle Composing
  - `W` $= 1$, `H` $= 1/2$, `Q` $= 1/4$, `E` $= 1/8$, `S` $= 1/16$, `T` $= 1/32$ y `X` $= 1/64$.
  - Sea una línea de la forma

    `/HH/QQQQ/XXXTXTEQH/W/HW/`
  - ¿Todos los compases tienen la misma duración?
- Mente Funcional
  - ¿Puedo hacer una transformación global?
  - ¿Puedo dividir esa transformación en partes?
  - Iteraciones implícitas – `map`, `fold`, ...
- Transformaciones puras primero – interacción después.

UNIVERSIDAD SIMÓN BOLÍVAR

Jingle Composing

ACM ICPC2009 Latin American Regionals

- Problem J – Jingle Composing
  - `W` $= 1$, `H` $= 1/2$, `Q` $= 1/4$, `E` $= 1/8$, `S` $= 1/16$, `T` $= 1/32$ y `X` $= 1/64$.
  - Sea una línea de la forma

    `/HH/QQQQ/XXXTXTEQH/W/HW/`
  - ¿Todos los compases tienen la misma duración?
- Mente Funcional
  - ¿Puedo hacer una transformación global?
  - ¿Puedo dividir esa transformación en partes?
  - Iteraciones implícitas – `map`, `fold`, ...
- Transformaciones puras primero – interacción después.
- ¿Habrá alguna librería que me ahorre trabajo?

UNIVERSIDAD SIMÓN BOLÍVAR

Son números si se pueden sumar...

- Representaré

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_k \cdot x^k$$

con la lista

$$[a_0, a_1, a_2, \cdots, a_k]$$

UNIVERSIDAD SIMÓN BOLÍVAR

Son números si se pueden sumar...

- Representaré

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_k \cdot x^k$$

con la lista

$$[a_0, a_1, a_2, \cdots, a_k]$$

- ¿Qué quiere decir ser `Num`?

UNIVERSIDAD SIMÓN BOLÍVAR

Son números si se pueden sumar...

- Representaré

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_k \cdot x^k$$

con la lista

$$[a_0, a_1, a_2, \cdots, a_k]$$

- ¿Qué quiere decir ser `Num`?
- ¿Qué quiere decir ser `Fractional`?

Todo para calcular generatrices...

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo generamos el Triángulo de Pascal?

Lazy evaluation FTW!

- La solución "obvia" – pero usa `(++)`

```
p0 = iterate (\r -> zipWith (+) (0:r) (r ++ [0])) [1]
```

¿Cómo generamos el Triángulo de Pascal?

Lazy evaluation FTW!

- La solución "obvia" – pero usa `(++)`

```
p0 = iterate (\r -> zipWith (+) (0:r) (r ++ [0])) [1]
```

- La solución que cabe en Twitter

```
p1 = iterate f [1]
  where f r@(c:cs)          = 1:z (+) r cs
        z g xs []           = xs
        z g (x:xs) (y:ys) = g x y:z g xs ys
```

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo generamos el Triángulo de Pascal?

Lazy evaluation FTW!

- La solución "obvia" – pero usa `(++)`

```
p0 = iterate (\r -> zipWith (+) (0:r) (r ++ [0])) [1]
```

- La solución que cabe en Twitter

```
p1 = iterate f [1]
  where f r@(c:cs)         = 1:z (+) r cs
        z g xs []          = xs
        z g (x:xs) (y:ys) = g x y:z g xs ys
```

¿Y si lo quiero imprimir más bonito?

UNIVERSIDAD SIMÓN BOLÍVAR